<u>Robotics – Exercise 4 – Blue Team Report</u>

Shlomi Ben-Shushan ID: 311408264
Yiftach Neuman ID: 208305359

Summery

In this report we will explain our implementation details of our algorithm that lets a Krembot forage in an unknown arena without using its current position. We will use this algorithm in the Adversarial Foraging tournament and compete with other classmates. Our team number is 1.

Algorithm Description

General Strategy:

Each forager will act according to one of the following three modes:

1. <u>Spiral Scanning Mode</u>: In this mode, the forager scans its surrounding for food in a spiral movement. The idea behind this mode, is that the food is spread uniformly in the arena, so there is the same good chance the there is a food nearby, without respect to the forager's starting position. This mode is stopped when the forager finds food (then it will change mode to **RTB** (i.e., Return-To-Base), or when it bumped into something (then it will change mode to **Move**), or when it finds the opponent's nest (then it will try to **block** it). This mode is profitable only when the forager is not carrying food (because the nest position only in one place), and when no-one interrupts it. Also, this mode is not profitable right after dropping food because probably the foragers will spend more time near the nest (because they drive towards it when they sense it), so if each forager will spirally scan for food immediately after dropping the food at the nest, we'll get a situation where the same positions are scanned over and over again. Hence, this mode is activated every 1.5-3 minutes and only among foragers that are not carrying food.
2. <u>Move Mode</u>: In this mode, the forager wander "close to randomly" in the arena. As before, once the forager finds food, it will change mode to **RTB**. It will make a **soft-turn** (i.e., with linear and angular speeds) when it senses a teammate ahead under the assumptions that robots are not big obstacles so there is no need to spend time on **hard-turn**, and if the teammate is not carrying food, it will act the same and by that we can spread robots in the arena (no need for multiple foragers in the same area), also if the teammate is carrying food, it is better to make way for him. The forager will make a hard-turn (i.e., with angular speed only) when it senses an obstacle that is not a robot ahead. The forager will **block** the opponent's nest if sense it.
3. <u>RTB Mode</u>: Stands for Return-To-Base. This mode is activated when the robot is carrying food. It is similar to **Move** mode with two main differences. First, the robot will drive to its team's nest when sense it. Second, the forager is **indifferent** to other robots. Even when carrying food, the forager will prefer **blocking** the opponent's nest due to the fact that it is an adversarial foraging so preventing the opponent from scoring two points pays off more than score another point.
4. <u>Block Nest Mode</u>: When sensing the opponent's nest, the forager will drive towards it and stop on it to block it from opponent's food-carrying foragers.

<u>Note 1</u>: In anytime, if the forager bumped into something, it would make a hard-turn, because of the understanding that any other maneuvers failed.

<u>Note 2</u>: in every mode, the top priority is to block the opponent's nest.

Forager's State: the following C++ enum defines the seven states we used.

```
enum State { spiralMove, spiralTurn, move, rtb, softTurn, hardTurn, blockNest };
```

The behavior of the forager in each state will be described in Action section.

Sense: in this section we will describe our usage in the robot's sensors.

1. **RGBA Cameras:** the forager uses 5 RGBA cameras (three in its front, and two on its sides) to identify colors and measure distances.
2. **Bumpers:** the forager uses the three bumpers in its front to sense collisions.

Interpretation: in this section we will describe how the forager interpret the sensing-data.

1. **Teammate Ahead:** when one of the three front RGBA cameras senses the team's color.
2. **Opponent Ahead:** when one of the three front RGBA cameras senses the opponent team's color.
3. **Robot Ahead:** when 1 or 2 takes place (have a `true` value).
4. **Obstacle Ahead:** when frontal distance is less than 10, and there is no robot ahead.
5. **Nest Ahead:** when the front RGBA camera senses friendly-nest's color.
6. **Nest to the Right:** when the right or the right-front RGBA cameras senses friendly-nest's color, and the nest is not ahead.
7. **Nest to the Left:** when the left or the left-front RGBA cameras senses friendly-nest's color, and the nest is not ahead.
8. **Opponent's Nest Ahead:** when the front RGBA camera senses opponent-nest's color.
9. **Opponent's Nest to the Right:** when the right or the right-front RGBA cameras senses opponent-nest's color, and the nest is not ahead.
10. **Opponent's Nest to the Left:** when the left or the left-front RGBA cameras senses opponent-nest's color, and the nest is not ahead.
11. **Opponent's Nest on Front:** 8, 9 or 10 occurred (have a `true` value).
12. **Bumped into something:** when one of the front bumpers is pressed.

Action: In this section we will describe each forager state we have defined, and the behavior of the robot (the actions it will take) in each of the state.

1. **Move:** drive straight until:
   a. Opponent's nest ahead → Switch to **Block-Nest** state.
   b. Found food → Switch to **RTB** state.
   c. Bumped into something → Evaluate which side is better to turn to and **hard**-turn.
   d. Teammate ahead → Evaluate which side is better to turn to and **soft**-turn.
   e. Obstacle ahead → Evaluate which side is better to turn to and **hard**-turn.
   Note that the evaluations in c. and d. are not the same.

2. **RTB:** drive straight until:
   a. Opponent's nest ahead → Switch to **Block-Nest** state.
   b. Food dropped → Switch to **Move** state.
   c. Nest (or base) nearby → Drive towards it.
   d. Bumped into something → Evaluate which side is better to turn to and **hard**-turn.
   e. Obstacle ahead → Evaluate which side is better to turn to and **hard**-turn.
   Note that the evaluations in c. and d. are not the same.

3. **Spiral-Move:** drive straight until:
    a. Opponent's nest ahead → Switch to **Block-Nest** state.
    b. Found food → Switch to **RTB** state.
    c. Bumped into something → Switch to **Move** (and Move state will handle it).
    d. Spiral Timer is finished → Make a 90 degree turn by switching to **Spiral-Turn** state.

4. **Soft-Turn:** drive with linear and angular speed until:
    a. Opponent's nest ahead → Switch to **Block-Nest** state.
    b. Found food → Switch to **RTB** state.
    c. Turning timer is finished → Switch to **Move**.

5. **Hard-Turn:** drive with angular speed only until:
    a. Turning-timer is finished → Switch to **RTB** if holds food, and to **Move** otherwise.

6. **Spiral-Turn:** drive with angular speed only until:
    a. Turning-timer is finished → Calculate the time for reaching the required distance to make a (squared) spiraling movement and switch to **Spiral-Move** state.

Setup() Pseudo-Code:

1. Register the robot for the group with `writeTeamColor()` function.
2. Define team, base, and opponent colors, w.r.t `foragingMsg.outColor` value.
3. Set initial state to Spiral-Move, and set other basic attributes.
4. Start sand-timers.

Loop() Pseudo-Code:

1. Read sensors.
2. Interpret sensing data.
3. If it is time to activate Spiraling-Mode, do it and restart timer.
4. Behave according to the current state, as described above.